TABLE 1

Pseudocode:

```
For each class file to be compiled
{
   For each method in the class containing bytecode instructions
   {
      Create storage to store actual mappings and native code addresses.
      Initialize stack mappings to empty and addresses to unknown.
      For each bytecode instruction from first to last
      {
         If there is a stack map stored for the actual bytecode instruction
         {
              If there is direct control flow from the previous instruction
            {
               Emit code to store all stack and set their stack mapping to 'stack'.
            }
            If there is no direct control flow from the previous instruction
            {
               Read stack layout from stack map in bytecode and set mappings to 'stack'
            }
            Set native code address for actual instruction.
         }
         If actual instruction is 'load constant'
         {
            Create a new 'constant' stack mapping .
         }
         If actual instruction is 'load local'
         {
            Create a new 'local' stack mapping.
         }
         If actual instruction is a stack manipulating instruction
              (one of pop, pop2, dup, dup_x1, dup_x2, dup2, dup2_x1, dup2_x2, or
           swap)
         {
            Duplicate and/or reorder stack mappings according to instruction.
         }
         If actual instruction is a jump or switch instruction
         {
            Emit code for the actual instruction using stack mapping information to
locate the
               arguments and native code addresses to get the actual destination address.
               Remove the mappings for the arguments.
              Emit code to store all stack values not used by this instruction and set their
           stack
                 mapping to 'stack'.
```

```
        }
         If actual instruction is any other instruction
        {
           Emit code for the actual instruction using stack mapping information to locate the
              arguments. Remove the mappings for the arguments and create a new mapping
              if the instruction results in a new value.
        }
      } /* For each bytecode instruction */
    } /* For each method */
  } /* For each class file */
```